DassFlow v1.0: a variational data assimilation software for river flows

Marc Honnorat, Joel Marin, Jerome Monnier, Xijun Lai

▶ **To cite this version:**

Marc Honnorat, Joel Marin, Jerome Monnier, Xijun Lai. DassFlow v1.0: a variational data assimilation software for river flows. [Research Report] RR-6150, INRIA. 2007. hal-00908169

HAL Id: hal-00908169
https://hal.science/hal-00908169

Submitted on 22 Nov 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

DassFlow v1.0: a variational data assimilation software for 2D river flows

Marc Honnorat — Joël Marin — Jérôme Monnier — Xijun Lai

N° 6150

Mars 2007

Thème NUM

ISSN 0249-6399 ISRN INRIA/RR--6150--FR+ENG

DassFlow v1.0: a variational data assimilation software for 2D river flows

Marc Honnorat* †, Joël Marin‡*, Jérôme Monnier§* , Xijun Lai¶ ‖

Thème NUM — Systèmes numériques
Projet MOISE

Rapport de recherche n° 6150 — Mars 2007 — 35 pages

Abstract: Dassflow is a computational software for river hydraulics (floods), especially designed for variational data assimilation. The forward model is based on the bidimensional shallow-water equations, solved by a finite volume method (HLLC approximate Riemann solver). It is written in Fortran 95. The adjoint code is generated by the automatic differentiation tool Tapenade. Thus, Dassflow software includes the forward solver, its adjoint code, the full optimization framework (based on the M1QN3 minimization routine) and benchmarks. The generation of new data assimilation twin experiments is easy. The software is interfaced with few pre and post-processors (mesh generators, GIS tools and visualization tools), which allows to treat real data.

Key-words: Variational data assimilation, parameter identification, river hydraulics, flood.

* LJK, BP 53, F-38041 Grenoble Cedex 9, France
† LJK - INPG & INRIA, <mailto:marc.honnorat@imag.fr>
‡ LJK - INRIA, <mailto:joel.marin@imag.fr>
§ LJK - INPG & INRIA, <mailto:jerome.monnier@imag.fr>
¶ Nanjing Institute of Geography , Limnology, Chinese Academy of Sciences
‖ <mailto:laixijin@gmail.com>

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

DassFlow v1.0: a variational data assimilation software for 2D river flows

Résumé : Dassflow est un code de calcul d'hydraulique fluviale (rivière, inondation) destiné à l'assimilation variationnelle de données. Le modèle direct est basé sur les équations de Saint-Venant (shallow water 2D) avec friction, et un schéma volumes finis explicite du type solveur de Riemann approché HLLC. Un grand nombre de conditions aux limites est implémenté dont celles de type caractéristique. Le code direct est écrit en Fortran 95 et destiné à être différencié automatiquement à l'aide du logiciel Tapenade. Ainsi, le logiciel Dassflow comporte le code direct, le code adjoint, le processus complet d'optimisation (basé sur la routine de minimisation locale L-BFGS, dénommée M1QN3) ainsi que des cas tests (benchmarks). La génération de nouvelles expériences jumelles en assimilation de données est facile. Ce code est interfacé avec divers outils de maillages (structuré, mixte triangles-quadrangles), SIG et visualiseurs (commerciaux et logiciels libres), ce qui permet de traiter des données réelles.

Mots-clés : Assimilation variationnelle de données, identification de paramètres, hydraulique fluviale, inondations.

Contents

1 Introduction **4**

2 The shallow water model (St Venant) **5**
2.1 The equations . . . 5
2.2 Boundary conditions . . . 5
2.2.1 Wall boundary conditions . . . 6
2.2.2 Inflow conditions . . . 6
2.2.3 Outflow conditions . . . 7

3 Variational data assimilation (4D-var) **8**
3.1 Observations and cost function . . . 9
3.2 Adjoint model . . . 10

4 Numerical implementation **12**
4.1 Forward code and numerical scheme . . . 13
4.1.1 The finite volume scheme . . . 13
4.1.2 The local 1D Riemann problem . . . 14
4.1.3 Stability condition . . . 15
4.1.4 Boundary conditions . . . 15
4.2 Adjoint code . . . 17
4.2.1 How to use the adjoint code? . . . 18
4.2.2 Adjoint code validation . . . 20

5 Benchmarks of the forward code **21**
5.1 Explicit solutions . . . 21
5.1.1 Water at rest . . . 22
5.1.2 Steady solution and Manning-Strickler formula . . . 22
5.1.3 Dambreak . . . 22
5.2 Real data test case: Moselle river . . . 23

6 Benchmarks for the full code (data assimilation) **24**
6.1 Identification of a local topography . . . 26
6.2 Identification of inflow discharge . . . 28
6.3 Identification of Manning coefficients . . . 29

A Characteristics boundary conditions **32**

B Riemann invariants and boundary conditions **33**

1 Introduction

The St-Venant equations (shallow water) can describe accurately river flows, in the main channel (1D geometry) or in the major bed (flood plain, 2D geometry). Nevertheless, in order to carry out a reliable numerical simulation, models require input parameters such as bed elevation, roughness coefficients in addition to initial and boundary conditions. Generally, these parameters are approximated (eg roughness coefficients), missing (eg inflow boundary conditions or initial conditions) or subjected to uncertainties. Moreover, the governing equations themselves may model ideal configurations, which not necessarily correspond to real configurations. Classicaly, expert-users of computational river flow softwares, see e.g. [**?**], must perform many simulations (hence spend time) in order to calibrate the numerical model related to a given configuration (trial-error tests). On the other hand, observations from in-situ gauge stations (eg water elevation) or remote sensing observations (satellite images, air photographs, video images) are very important sources of information on the flow state, but they are not qualitatively integrated into the simulation process. To improve the quality of the simulation, data assimilation methods combine in an optimal sense the information from the model and observated data. This allows to identify some parameter values consistent with reality, and /or initial conditions. Variational data assimilation method, see e.g. [**?**], [**?**], is based on the optimal control theory, [**?**], and it aims to fuse the dynamical model and observations by minimizing a cost function which measures the discrepancy between the simulation results and physical measurements. This method is operational in meterology and since more recently in oceanography. In river hydraulics, variational data assimilation methods have been used for the identification of model parameters in 1D channels, [**?**, **?**], and in 2D, [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**].
In real configurations, observations are available only in very small quantities. At best, water levels are measured at very few gauge stations (information very sparse in space); velocity measurements are even more scarce since they require complex human interventions.

DassFlow is a river hydraulics simulation software designed for variational data assimilation on simple hydraulic configurations for research purpose. The model is based on the bidimensional shallow-water equations, solved by the finite volume method using the HLLC approximate Riemann solver. The minimization procedure is based on the L-BFGS algorithm, [**?**]), which requires the computation of the gradient of the cost function. Since the forward code is written in Fortran 95, the adjoint code is generated by the automatic differentiation tool Tapenade [**?**] developped by the Tropics project-team at INRIA Sophia-Antipolis. DassFlow software is interfaced with few free and commercial pre and post-processors (SIG tools, mesh generators, visualization tools), which allows to performs computations with real data.

In Section 2, we present the Shallow Water model. In section 3, we present the Variational Data Assimilation method (4D-Var). In Section 4, we detail the finite volume scheme, then the way the adjoint code is automatically generated. In Section 5, we compare some solutions of the forward model with either explicit solutions or real measures (benchmarks

for the forward code). In Section 6, we consider twin experiments which show the capabilities of DassFlow (benchmarks for the data assimilation software). In appendix we detail characteristics boundary conditions and Riemann invariants.

The present research report treats of the mathematical and numerical features of Dassflow only. Concerning the code itself, we refer to the user manual [**?**] and the developer guide [**?**].

2 The shallow water model (St Venant)

2.1 The equations

The river hydraulics model is based on the bidimensional shallow water equations in their conservative formulation. The state variables are the water depth h and the local discharge $\mathbf{q} = h\mathbf{u}$, where $\mathbf{u} = (u, v)^T$ is the depth-averaged velocity vector. On a domain Ω and for a computational time interval $[0, T]$, the equations are:

$$\left\{\begin{array}{ll} \partial_t h \ + \ \mathrm{div}(\,\mathbf{q}\,) \ = \ 0 & \text{in } \Omega\times]\,0,T] \\ \partial_t \mathbf{q} \ + \ \mathrm{div}\left(\frac{1}{h}\,\mathbf{q}\otimes\mathbf{q}\right) \ + \ \frac{1}{2}g\nabla h^2 + \ gh\nabla z_b \ + \ g\frac{n^2\|\mathbf{q}\|_2}{h^{7/3}}\,\mathbf{q} \ = \ 0 & \text{in } \Omega\times]\,0,T] \\ h(0) \ = \ h_0, \qquad \mathbf{q}(0) \ = \ \mathbf{q}_0 & \\ + \text{ boundary conditions (see section 2.2)} & \end{array}\right. \tag{1}$$

where g is the magnitude of the gravity, z_b the bed elevation, n the Manning roughness coefficient, h_0 and q_0 the initial conditions for the state variables. The quantity $c = \sqrt{gh}$ denotes the local wave celerity.
It is important to notice that if the bed elevation is constant ($\nabla z_b = 0$) and without the friction term, (1) is an hyperbolic system.

Moreover, a transport equation related to a tracer Φ is considered:

$$\partial_t\, h\Phi \ + \ \mathrm{div}(\,\mathbf{q}\Phi\,) \ = \ s_f \tag{2}$$

with initial and boundary conditions. s_f is the tracer source term.

2.2 Boundary conditions

We split the boundary Γ of the domain Ω in three parts (see figure 2.1). We denote by:

- Γ_{in} the part where the flow is incoming (type INFLOW)
- Γ_{out} the part where the flow is outgoing (type OUTFLOW)
- Γ_{wall} the part where it is a closed boundary (type WALL)

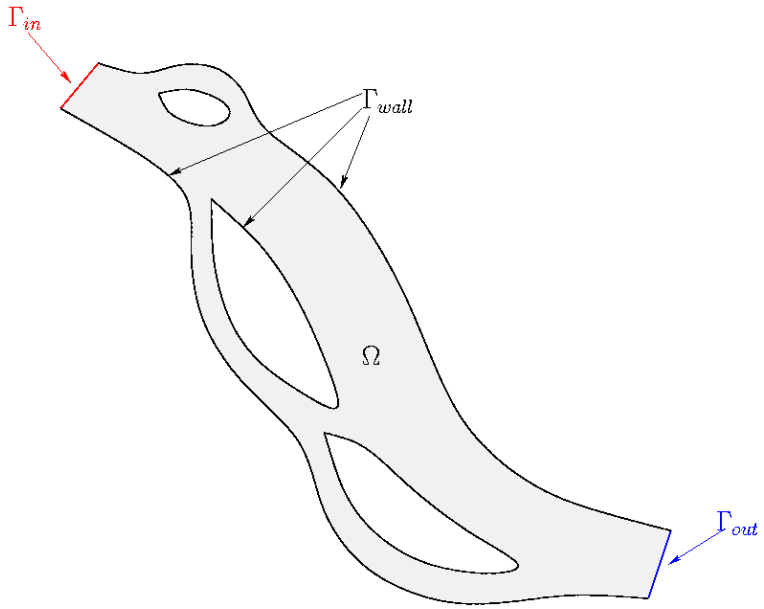


Figure 2.1: The simulation domain and its boundaries

The user has a large choice of boundary conditions, depending on the boundary condition type.

2.2.1 Wall boundary conditions

On Γ_{wall}, we consider a slip condition:

$$\mathbf{u}\cdot\mathbf{n}|_{\Gamma_c}(t) \;=\; 0\,, \qquad \frac{\partial h}{\partial \mathbf{n}}|_{\Gamma_c}(t) \;=\; 0 \qquad \forall\, t \in]0,T]$$

2.2.2 Inflow conditions

Few boundary conditions are possible at inflow ie on Γ_{in}.

Discharge imposed

We have:

$$(\mathbf{q}\cdot\mathbf{n})|_{\Gamma_{in}} = \; -q^{in}(t)\,, \qquad \frac{\partial h}{\partial \mathbf{n}}|_{\Gamma_{in}}(t) \;=\; 0 \qquad \forall\, t \in]0,T]$$

Incoming characteristics

The characteristic variables of the system are: $w_1 = \mathbf{u}\cdot\mathbf{n} + \sqrt{\frac{g}{h_0}}h$, $w_2 = \mathbf{u}\cdot\tau$ and $w_3 = \mathbf{u}\cdot\mathbf{n} - \sqrt{\frac{g}{h_0}}h$, associated to the eigenvalues $\lambda_1 = \mathbf{u}_0\cdot\mathbf{n}+c$, $\lambda_2 = \mathbf{u}_0\cdot\tau$ and $\lambda_3 = \mathbf{u}_0\cdot\mathbf{n}-c$ respectively. Let us recall that: $c = \sqrt{gh_0}$.

We detail more concerning the characteristics in appendix A.
The characteristic w_k is incoming if $\lambda_k < 0$ and outgoing otherwise. The characteristics boundary conditions are implemented as follows:

. For each point of the boundary, we compute the three eigenvalues λ_k.

. For each λ_k,

  – if $\lambda_k \geq 0$, then w_k is incoming; thus it is evaluated from external data,
  – if $\lambda_k < 0$, then w_k is outgoing; thus it is evaluated from the internal simulation domain.

. Then, the model variables can be retrieved as follows:

  – $\mathbf{u}\cdot\mathbf{n}^{bnd} = \frac{w_1+w_3}{2}$
  – $\mathbf{u}\times\mathbf{n}^{bnd} = w_2$
  – $h^{bnd} = \sqrt{\frac{h_0}{g}}\frac{w_1-w_3}{2}$

2.2.3 Outflow conditions

Few boundary conditions are possible at outflow ie on Γ_{out}.

Homogeneous Neumann

$$\frac{\partial h}{\partial \mathbf{n}}|_{\Gamma_{out}}(t) \;=\; 0\,,\; \frac{\partial \mathbf{q}}{\partial \mathbf{n}}|_{\Gamma_{out}}(t) \;=\; 0 \qquad \forall\, t \in]0,T]$$

Water elevation prescribed

$$h|_{\Gamma_{out}}(t) \;=\; z_{out}(t) - z_b|_{\Gamma_{out}},\; \frac{\partial(\mathbf{u}\cdot\mathbf{n}+2c)}{\partial \mathbf{n}}|_{\Gamma_{out}}(t) \;=\; 0\;\forall\, t \in]0,T]$$

This boundary condition should be applied only in case of sub-critical flow. We refer to appendix B for a justification of the relation $\frac{\partial(\mathbf{u}\cdot\mathbf{n}+2c)}{\partial \mathbf{n}}|_{\Gamma_{out}}(t) \;=\; 0$

Incoming characteristics

This case is similar to those related to the inflow boundary Γ_{in}.

Rating curve

A rating curve is a relation between the normal discharge $\mathbf{q}\cdot\mathbf{n}$ and the water height h : $\mathbf{q}\cdot\mathbf{n} = f(h)$. Such a relation can be used to specify outflow boundary conditions if combined with the relation: $\frac{\partial(U+2\sqrt{gh})}{\partial n} = 0$.

To this end, we solve the following non-linear system (see figure 2.2):

$$\begin{cases} \mathbf{q}\cdot\mathbf{n} = f(h) \\ \mathbf{u}\cdot\mathbf{n} + 2\sqrt{gh} = \left(\mathbf{u}\cdot\mathbf{n} + 2\sqrt{gh}\right)_{int} \end{cases} \tag{3}$$

where $\left(\mathbf{u}\cdot\mathbf{n} + 2\sqrt{gh}\right)_{int}$ is computed from the interior of the domain.

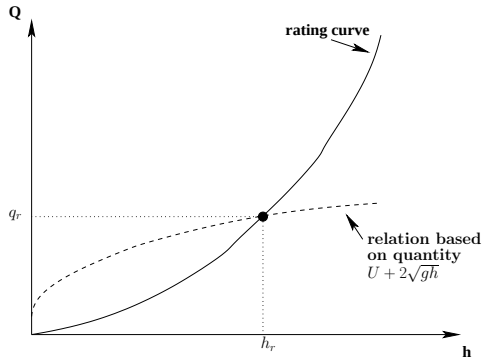


Figure 2.2: Outflow BC: Rating curve and $U + 2\sqrt{gh} = 0$

3 Variational data assimilation (4D-var)

The DassFlow software is designed for variational data assimilation (classicaly called 4D-var method). Variational data assimilation is based on the optimal control theory, see [**?**], [**?**]. Briefly, this method consists to compute a control vector value $\mathbf{k}$ minimizing a cost function which measures the discrepancy between the computed variable and available data.
In our case the control vector $\mathbf{k}$ can include the initial condition, boundary conditions, manning coefficient and bed elevation. For example, if inflow discharge is prescribed and water elevation is prescribed, we get:

$$\mathbf{k} = (h_0, \mathbf{q}_0, q_{in}, z_{out}, n, z_b)^T$$

If we impose incoming characteristics for both inflow and outflow boundaries, we get:

$$\mathbf{k} = (h_0, \mathbf{q}_0, [w_1^{in}, w_2^{in}, w_3^{in}], [w_1^{out}, w_2^{out}, w_3^{out}], z_{out}, n, z_b)^T$$

We set below the observation function $H(\mathbf{k}; h, \mathbf{q})$ and the cost function $J(\mathbf{k}) = H(\mathbf{k}; h, \mathbf{q})$ which measures the discrepancy between the state of the system (computed variable) and observations. Then the optimal control problem we solve reads:

$$Min_{\mathbf{k}}\ J(\mathbf{k}) \tag{4}$$

where $(h, \mathbf{q})$ is the solution of the forward model (1).

This optimization problem is solved numerically by a descent algorithm. Thus, we need to compute the gradient of the cost function. This is classicaly done by introducing the adjoint model.

Let us point out that initial conditions, boundary conditions, manning coefficients and bed elevation are only potential control variables. In practice, one manages to identify very few of them only at the same time.

3.1 Observations and cost function

Given observations, we define the cost function J which measures the discrepancy between the computed variable (state of the system) and the available data.

In the present version, we assume that water depth and discharge are available at some point and some time instant. In the forthcoming version of DassFlow, we will assume to have some extra observations such as trajectories at surface (lagrangian data).

Given such (eulerian) observations, we define the cost function as the sum of three terms.

- The first term measures the discrepancy between observation data and computed state variable :

$$J_{obs}(\mathbf{k}) = \frac{1}{2}\left(\int_0^T \|C_h\ h(\mathbf{k},t) - h^{obs}(t)\|^2 + \int_0^T \|C_{\mathbf{q}}\ \mathbf{q}(\mathbf{k},t) - \mathbf{q}^{obs}(t)\|^2\right)$$

  where C_h and $C_{\mathbf{q}}$ are restriction operators.

- The second term measures the discrepancy of the flux:

$$J_{flux}(\mathbf{k}) = \frac{1}{2}\int_0^T \left\|G(\mathbf{k},t) - G^{obs}(\mathbf{k},t)\right\|^2$$

with

$$G(\mathbf{k},t) - G^{obs}(\mathbf{k},t) = \frac{\Delta t}{|K_i|} \sum_{j=1}^{4} T_{ij}^{-1} \left(\widetilde{G}_1(U_L, U_R) - \widetilde{G}_1(\tilde{U}_L, U_R) \right)$$

with $\tilde{U}_L = [h^{obs},\ h^{obs}u,\ h^{obs}v]^T$ and $\widetilde{G}_1(U_L, U_R)$ is the first component of the flux computed by the HLLC solver (see section 4.1), ie the discharge through the edge between cell L and cell R. That means that this cost function measures the difference on the sum of the flux through all edges of the observed cell (see figure 3.3).

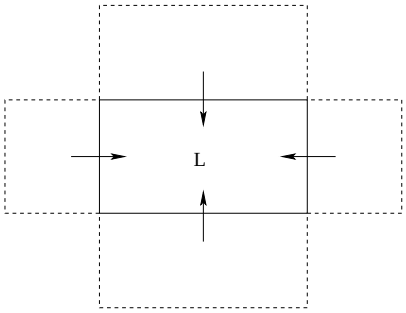


Figure 3.3: Flux through all edges of cell L

- The third term is a classical regularization term. For example, if we control inflow discharge, we set:

$$J_p(\mathbf{k}) = \frac{1}{2} \| \partial_{tt}^2 q_{in} \|^2$$

Finally, the cost function writes :

$$J(\mathbf{k}) = \alpha_{obs} J_{obs}(\mathbf{k}) + \alpha_{flux} J_{flux}(\mathbf{k}) + \alpha_p J_p(\mathbf{k}) \tag{5}$$

where α_{obs}, α_{flux} and α_p are scaling coefficients which must be setted by user.

3.2 Adjoint model

In order to compute efficiently all partial derivatives of the cost function $J(\mathbf{k})$ with respect to the components of the control vector $\mathbf{k}$, we introduce the adjoint model, [**?**].[**?**].

Given the state of the system $(h, \mathbf{q})$, solution of the forward model, we compute $(h^*, \mathbf{q}^*)$, solution of the adjoint model:

$$\left\{\begin{array}{l} \partial_t\, h^* - \frac{1}{h^2}\big[(\mathbf{q}\cdot\nabla)\,\mathbf{q}^*\big]\cdot\mathbf{q} + gh\,\mathrm{div}(\mathbf{q}^*) \\ \quad - g\,\mathbf{q}^*\cdot\nabla z_b + \frac{7}{3}g\frac{n^2\|\mathbf{q}\|}{h^{7/3}}\mathbf{q}\cdot\mathbf{q}^* \;=\alpha_{obs}\; C_h^T\Big(C_h\,h - h^{obs}\Big) \\ \qquad\qquad\qquad\qquad + \alpha_{flux}\,\big[D_{\mathbf{k}}(G-G^{obs})\big]^T\,(G(\mathbf{k},t)-G^{obs}(\mathbf{k},t)) \\ \partial_t\,\mathbf{q}^* + \nabla h^* + \frac{1}{h}(\mathbf{q}\cdot\nabla)\,\mathbf{q}^* + \frac{1}{h}(\nabla\mathbf{q}^*)^T\,\mathbf{q} \\ \quad - g\frac{n^2\|\mathbf{q}\|}{h^{7/3}}\mathbf{q}^* - g\frac{n^2}{h^{7/3}\|\mathbf{q}\|}(\mathbf{q}\otimes\mathbf{q})\mathbf{q}^* \;=\alpha_{obs}\; C_{\mathbf{q}}^T\Big(C_{\mathbf{q}}\,\mathbf{q} - \mathbf{q}^{obs}\Big) \\ h^*(T) \;=\; 0\,, \quad \mathbf{q}^*(T) \;=\; 0\,. \end{array}\right. \tag{6}$$

Let us point out that the adjoint model is reverse in time, and it includes the observations. Concerning its boundary conditions, for example in the case of inflow discharge prescribed at inflow and water elevation prescribed at outflow, we get:

$$\begin{array}{ll} \underline{\text{C.L.}}: & \mathbf{q}^*|_{\Gamma_{in}} \;=\; 0\,, \quad (\mathbf{q}^*\cdot\mathbf{n})|_{\Gamma_{wall}} \;=\; 0\,, \quad (\mathbf{q}^*\cdot\tau)|_{\Gamma_{out}} \;=\; 0\,, \\ & (\partial_{\mathbf{n}}h^*)|_{\Gamma_{in}\cup\Gamma_{wall}} \;=\; 0\,, \quad \Big(h^* + 2(\mathbf{u}\cdot\mathbf{n})(\mathbf{q}^*\cdot\mathbf{n})\Big)|_{\Gamma_{out}} \;=\; 0\,. \end{array} \tag{7}$$

Given the state $(h, \mathbf{q})$ and the adjoint state $(h^*, \mathbf{q}^*)$, one obtains the following expression of the partial derivatives of the cost function (independent of the derivative of the state):

$$\begin{array}{rcl} \dfrac{\partial j}{\partial h_0}(\mathbf{k}) & = & -h^*(0)\,, \qquad \dfrac{\partial j}{\partial \mathbf{q}_0}(\mathbf{k}) \;=\; -\mathbf{q}^*(0)\,, \\ \dfrac{\partial j}{\partial z_b}(\mathbf{k}) & = & -\displaystyle\int_0^T \mathrm{div}\Big(gh(t)\mathbf{q}^*(t)\Big)\,dt\,, \\ \dfrac{\partial j}{\partial n}(\mathbf{k}) & = & 2\,g\,n\displaystyle\int_0^T \big\|\mathbf{u}(t)\big\|\,h(t)^{-\frac{1}{3}}\big\langle\mathbf{u}(t),\mathbf{q}^*(t)\big\rangle_\Omega\,dt\,, \\ \dfrac{\partial j}{\partial q_{in}}(\mathbf{k}) & = & \alpha_p\partial_{tt}^{2\,T}\partial_{tt}^2 q_{in} - h^*|_{\Gamma_q}, \\ \dfrac{\partial j}{\partial z_{out}}(\mathbf{k}) & = & \Big[(\mathbf{q}^*\cdot\mathbf{n})\Big(c^2 - (\mathbf{u}\cdot\mathbf{n})^2\Big)\Big]\,|_{\Gamma_z}\,. \end{array} \tag{8}$$

A single integration of the direct model (1) followed by a single integration backward in time of the adjoint model (6)-(7) allow to compute all components of the gradient of the cost function (see figure 3.4).

The optimal control problem (4) is solved using a local descent algorithm, more precisely the L-BFGS algorithm (a quasi-Newton method), implemented in the M1QN3 routine, see [**?**, **?**]. Thus, these partial derivatives are used as input to the minimization algorithm

M1QN3. The global optimization process is represented in Fig. 3.4.

In practice, we don't implement the adjoint model presented above but we compute the partial derivatives directly by differentiating the forward code using an automatic differentiation tool, see next section.

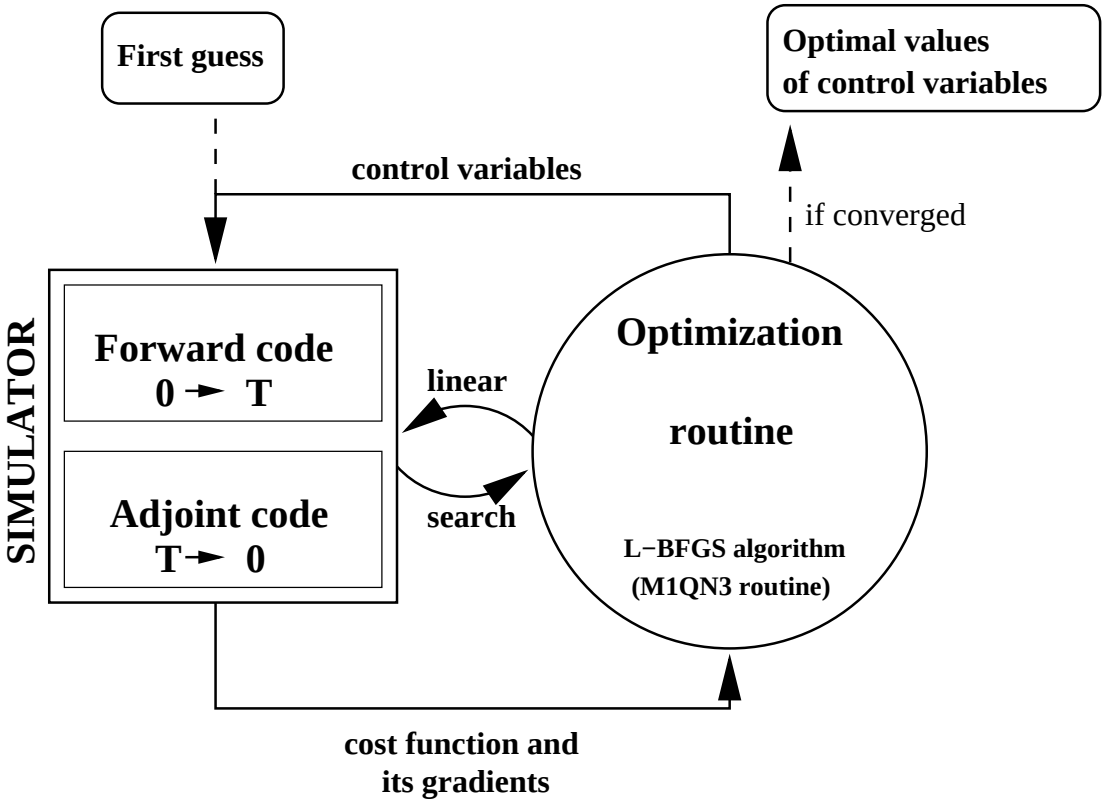


Figure 3.4: Variational data assimilation process

4 Numerical implementation

In this section, we describe on one hand the finite volume scheme solving the forward model and its implementation, on the other hand the automatic differentiation of the forward code in order to obtain the partial derivatives of J. For more details concerning the numerical implementation of the finite volume scheme, one can consult [**?**], [**?**].

4.1 Forward code and numerical scheme

The forward (or direct) model (1) is discretized using by a finite volume method based on the approximate Riemann solver HLLC, see e.g. [**?**]. The 2D mesh can be a mix of triangular and quadrangular cells.

4.1.1 The finite volume scheme

Equations (1) are rewritten as follows :

$$\partial_t U + \partial_x(G(U)) + \partial_y(H(U)) - (S_g(U) + S_f(U)) = 0, \quad (x,y) \in \Omega_2 \ , \quad t \in (0,T)$$

where $U = [h, q_x, q_y]^T$,

$$G(U) = [q_x, q_x^2/h + gh^2/2, q_x q_y/h]^T$$
$$H(U) = [q_y, q_x q_y/h, q_y^2/h + gh^2/2]^T$$
$$S_g(U) = [0, gh\partial_x z_b, gH\partial_y z_b]^T$$

z_b is the topography term, S_f is the 2D friction term (Manning law) and g is the magnitude of the gravity.

For the finite volume K_i of the mesh, we define the mean value of the state variable U:

$$U_i = \frac{1}{|K_i|} \int_{K_i} U \, d\Omega \, , \tag{9}$$

where $|K_i|$ denotes the surface of the cell. We integrate the equation over K_i, we use the divergence theorem, we use the rotational invariance property of the equations [**?**], then we obtain:

$$\int_{K_i} \partial_t U \ dx \ + \ \sum_{j=1}^{N_i} \int_{E_{ij}} T_{ij}^{-1} G(T_{ij} U) \ ds \ - \ \int_{K_i} (S_g(U) + S_f(U)) \ dx \ = \ 0 \tag{10}$$

where N_i denotes the number of faces of the cell K_i (3 or 4), E_{ij} is the cell interface and T_{ij} is the rotation of angle θ_{ij} between the normal to E_{ij} and the horizontal plane, Fig. 4.5:

$$T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\ \theta & sin\ \theta \\ 0 & -sin\ \theta & cos\ \theta \end{pmatrix}$$

Using the forward Euler scheme for temporal discretization, the equation becomes:

$$U_i^{m+1} \ = \ U_i^m \ - \ \frac{\Delta t}{|K_i|} \sum_{j=1}^{N_i} T_{ij}^{-1} G^m(T_{ij} U_i) \ + \ \Delta t \ S_g^m(U_i) \ + \ \Delta t \ \widetilde{S_f^{m+1}}(U_i) \tag{11}$$

where $m = 0, ..., T/\Delta t$ is the time index, Δt is the time step used for the 2D model integration. $\widetilde{S_f^{m+1}}$ denotes the friction term treated semi-implicitely in time.

The difficulty is to compute the numerical flux $G^m(T_{ij}U_i) = \int_{E_{ij}} G(T_{ij}U)\ ds$.

The rotational invariance property of the equations allows to reduce the 2D shallow-water problem to a sum of 1D local Riemann problems. These 1D Riemann problems are solved using the HLLC solver which consists to approximate the 1D flux G, see e.g. [**?**] and [**?**].
Moreover, the topography term S_g is included in the HLLC solver (see section 4.1.2). Thus, (11) writes :

$$U_i^{m+1} \ = \ U_i^m \ - \ \frac{\Delta t}{|K_i|} \sum_{j=1}^{N_i} T_{ij}^{-1} \tilde{G}^m(T_{ij}U_i) \ + \ \Delta t \ \widetilde{S_f^{m+1}}(U_i) \tag{12}$$

Only the friction term is treated outside the HLLC numerical flux. It is discretized semi-implicitely in time as follows:

$$\widetilde{S^{m+1}}(U_i) = \begin{pmatrix} 0 \\ \left(-g\frac{n^2\|\mathbf{u}\|}{h^{4/3}}\right)^m (q_x)^{m+1} \\ \left(-g\frac{n^2\|\mathbf{u}\|}{h^{4/3}}\right)^m (q_y)^{m+1} \end{pmatrix}$$

where n is the manning coefficient.

4.1.2 The local 1D Riemann problem

We detail below the computation of an approximation of the 1D flux G. This is done as follows.

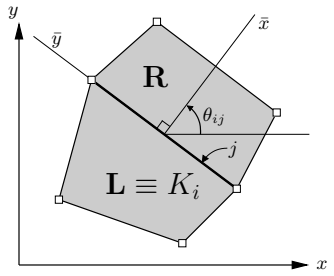


Figure 4.5: Adjacent cells and use of the rotational invariance property of the equations

1) First, we compute the state variable in local coordinates in the two cells K_L and K_R, Fig. 4.5, using the rotation: $V_i = T_{ij}U_i$. Subscripts L and R denote cells respectively to the left and to the right of the interface.

2) If $V = [H, Q_{\vec{n}}, Q_{\vec{\tau}}]^T$ where (τ, n) denotes the tangential and normal unit vectors respectively, we have the local normal 2D flux:

$$G(V) = [Q_{\vec{n}}, Q_{\vec{n}}^2/H + gH^2/2, Q_{\vec{n}}Q_{\vec{\tau}}H]^T$$

In the homogeneous case ($S_g = 0$), the 2D finite volume solver consists to solve, for each edge, the following local 1D Riemann problem:

$$\partial_t V \ + \ \partial_{\mathbf{n}} G(V) \ = \ 0$$

with $V(x,0) = V_L$ if $x_{\vec{n}} < 0$; $V(x,0) = V_R$ if $x_{\vec{n}} > 0$. The normal flux $G(V)$ is computed using the HLLC solver, [**?**].
Finally, we obtain the new solution U_i^{m+1} using (11).

In the non-homogeneous case ($S_g \neq 0$), the topography term is included in the flux term, using the following modification of $G_2(V)$ proposed by LeVeque [**?**] :

$$\tilde{G}(V) = G(V) + \begin{pmatrix} 0 \\ \frac{1}{2}g(h_L + h_R)(z_{bR} - z_{bL}) \\ 0 \end{pmatrix}$$

4.1.3 Stability condition

Since we use the forward Euler scheme in time, we do not have linear system to solve but the time step must respect a stability condition (CFL type). In the present case, this CFL stability condition is, see [**?**]:

$$\text{CFL} \ = \ \Delta t \, \frac{\max\left(\|\mathbf{u}\| + c\right)}{\min(d_{L,R})} \leq 1 \ , \tag{13}$$

where $d_{L,R}$ is the distance between the cell L and the center of the edge separing it from cell R.

4.1.4 Boundary conditions

In this section, we describe the way we impose the different boundary conditions described in section 2.2.

Let us consider a "boundary cell" L, ie it has no neighboor through its edge i, (see figure 4.6). In the context of a finite volume scheme, we have to evaluate the flux through the edge i. Thus, it becomes natural to define a "ghost-cell" R, for each boundary cell L.

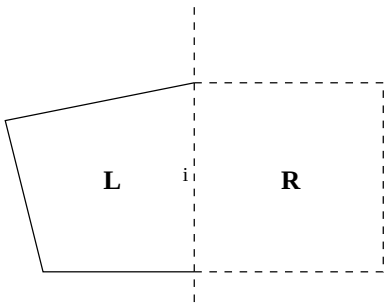


Figure 4.6: Internal cell L at boundary and ghost cell R

In order to impose a boundary condition (e.g. inflow dicharge, incoming characteristics etc), we set a particular value to $U_R = (h_R, u_R, v_R)^T$ such that the boundary condition is imposed when solving the local Riemann problem through the edge i (ie by computing the numeric flux $\tilde{G}(U_L, U_R)$). Let us detail the values U_R required for each type of boundary conditions.

Wall conditions On Γ_{wall}, we consider slip type conditions, then:

- Homogeneous Neumann condition on h is imposed by setting $h_R = h_L$
- Slip condition on $\mathbf{u}$ is imposed by setting $u_R = -u_L$ and $v_R = v_L$

Inflow conditions On Γ_{in}, we impose one of the two following boundary conditions.

. Inflow discharge
The inflow discharge is imposed (unit in $m^3 s^{-1}$). This corresponds to the discharge through the whole boundary and it is distributed in each cell such that the normal velocity u_L is constant along the boundary.

  – Homogeneous Neumann on h are prescribed by setting $h_R = h_L$.
  – The inflow discharge is prescribed by setting $u_R = -\frac{q^{in}(t)}{\mathcal{A}^n}$, where $\mathcal{A}^n$ is the boundary wet section area, and $v_R = v_L$.

. Incoming characteristics
Once incoming and outgoing characteristics w_1, w_2 and w_3 are computed in the ghost cell as described in appendix A, we set:

  – $h_R = \frac{w_1 - w_3}{2c}$
  – $u_R = \frac{w_1 + w_3}{2h_R}$
  – $v_R = w_2$

Outflow conditions On Γ_{out}, we impose one of the four following boundary conditions.

. Water elevation

- We impose $h_R = z_{prescribed} - (z_b)_R$, where $(z_b)_R$ is the bathymetry in the ghost-cell.
- $v_R = v_L$
- We impose $(u_R + 2\sqrt{gh_R}) = (u_L + 2\sqrt{gh_L})$, that is $u_R = (u_L + 2\sqrt{gh_L} - 2\sqrt{gh_R})$.

. Homogeneous Neumann We impose $h_R = h_L$, $u_R = u_L$ and $v_R = v_L$.

. Incoming characteristics
Once incoming and outgoing characteristics w_1, w_2 and w_3 are computed in the ghost cell as described in appendix A, we set:

- $h_R = \frac{w_1 - w_3}{2c}$
- $u_R = \frac{w_1 + w_3}{2h_R}$
- $v_R = w_2$

. Rating curve

- As described in section 2.2.3, we solve:

$$\begin{cases} \mathbf{q}_R = f(h_R) \\ \mathbf{u}_R + 2\sqrt{gh_R} = \mathbf{u}_L + 2\sqrt{gh_L} \end{cases} \tag{14}$$

- In addition we impose: $v_R = v_L$

4.2 Adjoint code

In practice, there exists three approaches to compute the adjoint state variable, then the gradient of the cost function.

1. The discretized continuous gradient can be obtained from the adjoint model (6)-(7) discretized using an appropriate numerical scheme which is then implemented.
2. The discrete gradient can be obtained from the adjoint of the direct numerical scheme and its implementation.
3. The "computational gradient" is obtained from the differentiation of the forward code directly.

This last approach ensures a better consistency between the computed cost function (including all types of errors -errors of discretization, rounding errors, iterative algorithms etc-) and its gradient since it is the computed cost function which is differentiated. A large part of this extensive task can be automated using algorithmic differentiation, see [**?**]. In the case of DassFlow, the direct code is written in Fortran 95 and it is derived using the automatic differentiation tool Tapenade [**?**].

4.2.1 How to use the adjoint code?

We describe how to define the direct code, then what is the response of the adjoint code automatically generated and finally how to use it.
Let $\mathcal{K}$ be the space of control variables and $\mathcal{Y}$ the space of the forward code response. In the case of DassFlow, we have :

$$\mathbf{k} = (y_0, q_{in}, z_{out}, n, z_b,)^T \quad \text{and} \quad Y = \big(y, j\big)^T$$

Let us point out that we include both the state and the cost function in the response of the forward code.

We can represent the direct code as the operator $\mathcal{M} : \mathcal{K} \longrightarrow \mathcal{Y}$, see figure 4.7.

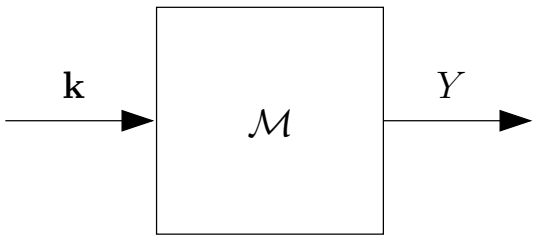


Figure 4.7: Representation of the direct model.

The tangent model becomes $\frac{\partial \mathcal{M}}{\partial \mathbf{k}}(\mathbf{k}) : \mathcal{K} \longrightarrow \mathcal{Y}$. It takes as input variable a perturbation of the control vector $d\mathbf{k} \in \mathcal{K}$, then it gives the variation $dY \in \mathcal{Y}$ as output variable, see figure 4.8) :

$$dY = \frac{\partial \mathcal{M}}{\partial \mathbf{k}}(\mathbf{k}) \cdot d\mathbf{k}$$

The adjoint model is defined as the adjoint operator of the tangent model. This can be represented as follows: $\big(\frac{\partial \mathcal{M}}{\partial \mathbf{k}}(\mathbf{k})\big)^* : \mathcal{Y}' \longrightarrow \mathcal{K}'$. It takes $dY^* \in \mathcal{Y}'$ an input variable and provides the adjoint variable $d\mathbf{k}^* \in \mathcal{K}'$ at output, see figure 4.9:

$$d\mathbf{k}^* = \left(\frac{\partial \mathcal{M}}{\partial \mathbf{k}}(\mathbf{k})\right)^* \cdot dY^*$$

Now, let us make the link between the adjoint code and the "computational gradient".

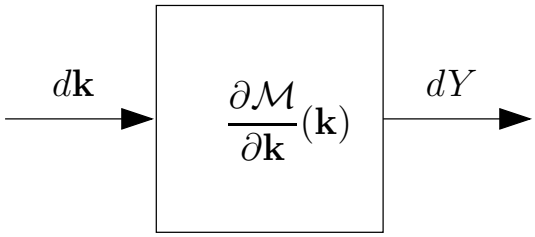


Figure 4.8: Representation of the tangent model.

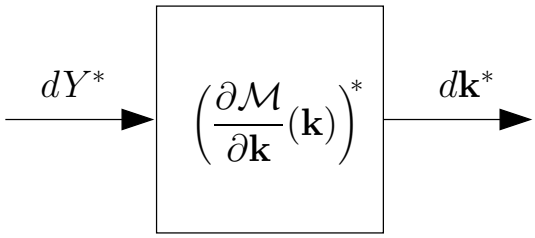


Figure 4.9: Representation of the adjoint model.

By definition of the adjoint, we have :

$$\left\langle \left(\tfrac{\partial \mathcal{M}}{\partial \mathbf{k}}\right)^* \cdot dY^*,\, d\mathbf{k} \right\rangle_{\mathcal{K}' \times \mathcal{K}} = \left\langle dY^*,\, \left(\tfrac{\partial \mathcal{M}}{\partial \mathbf{k}}\right) \cdot d\mathbf{k} \right\rangle_{\mathcal{Y}' \times \mathcal{Y}} \tag{15}$$

or, using the relations presented above:

$$\left\langle d\mathbf{k}^*,\, d\mathbf{k} \right\rangle_{\mathcal{K}' \times \mathcal{K}} = \left\langle dY^*,\, dY \right\rangle_{\mathcal{Y}' \times \mathcal{Y}} . \tag{16}$$

If we set $dY^* = (0,1)^T$ and by denoting the perturbation vector $d\mathbf{k} = (\delta y_0, \delta n, \delta z_b, \delta q^{in})^T$, we obtain:

$$\left\langle \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} dy^* \\ dJ^* \end{pmatrix} \right\rangle_{\mathcal{Y}' \times \mathcal{Y}} = \left\langle \begin{pmatrix} \delta y_0^* \\ \delta n^* \\ \delta z_b^* \\ \delta q^{in*} \\ \delta z_{out}^* \end{pmatrix}, \begin{pmatrix} \delta y_0 \\ \delta n \\ \delta z_b \\ \delta q^{in} \\ \delta z_{out} \end{pmatrix} \right\rangle_{\mathcal{K}' \times \mathcal{K}}$$

Moreover, we have by definition:

$$dJ = \tfrac{\partial J}{\partial y_0}(\mathbf{k}) \cdot \delta y_0 + \tfrac{\partial J}{\partial n}(\mathbf{k}) \cdot \delta n + \tfrac{\partial J}{\partial z_b}(\mathbf{k}) \cdot \delta z_b + \tfrac{\partial J}{\partial q_{in}}(\mathbf{k}) \cdot \delta q_{in} + \tfrac{\partial J}{\partial z_{out}}(\mathbf{k}) \cdot \delta z_{out}$$

Therefore, the adjoint variable $d\mathbf{k}^*$ (output of the adjoint code with $dY^* = (0,1)^T$) corresponds to the partial derivatives of the cost function J:

$$\begin{array}{lll} \frac{\partial J}{\partial y_0}(\mathbf{k}) = y_0^* & \frac{\partial J}{\partial n}(\mathbf{k}) = n^* & \\ \frac{\partial J}{\partial z_b}(\mathbf{k}) = z_b^* & \frac{\partial J}{\partial q_{in}}(\mathbf{k}) = q_{in}^* & \frac{\partial J}{\partial z_{out}}(\mathbf{k}) = z_{out}^* \end{array}$$

In summary, in order to compute the "computational gradient" (partial derivatives of the cost function J using differentiation of the forward code), first, one runs the direct code then one runs the adjoint code with $dY^* = (0,1)^T$ as input.

Automatic differentiation TAPENADE [**?**] is an automatic differentiation tool for Fortran programs. It is devopped by the TROPICS team [**?**] at INRIA Sophia-Antipolis. TAPENADE works using source code transformation : it builds the tangent and/or adjoint code automatically from the direct code written in Fortran. One refers also to [**?**] to get a detailed description of how automatic differentiation works.

4.2.2 Adjoint code validation

We describe below how we check the validity of the adjoint code. Classicaly, we check that it is actually the adjoint of the tangent linear code (scalar product test) and that it computes correctly the partial derivative of the cost function (gradient test).

Scalar product test

The objective of this test is to check if the adjoint code is actually the adjoint of the tangent linear code. In other words, we check the relation (15) :

- Given an arbitrary $d\mathbf{k} \in \mathcal{K}$, we compute using the tangent linear code :
$$dY = \left(\frac{\partial \mathcal{M}}{\partial \mathbf{k}}\right) \cdot d\mathbf{k}$$
- Given an arbitrary $dY^* \in \mathcal{Y}$, we compute using the adjoint code :
$$d\mathbf{k}^* = \left(\frac{\partial \mathcal{M}}{\partial \mathbf{k}}\right)^* \cdot dY^*$$
- Then, we compute the following scalar products :
  - $sp_1 = \left\langle dY^*, dY \right\rangle_{\mathcal{Y}}$
  - $sp_2 = \left\langle d\mathbf{k}^*, d\mathbf{k} \right\rangle_{\mathcal{K}}$
- And we check if $sp_1 = sp_2$ or not.

Figure 4.10 (b) shows an example of the scalar product test.

Gradient test

The objective of this test is to check if the adjoint variables dX^* computed by the adjoint code correspond to the partial derivatives of the cost function.
The Taylor expansion of the cost function j at $\mathbf{k}$ for a small perturbation $\alpha\,\delta\mathbf{k}$ (where $\alpha \in \mathbb{R}^+$) writes :

$$j(\mathbf{k} + \alpha\,\delta\mathbf{k}) \;=\; j(\mathbf{k}) + \alpha\;\frac{\partial j}{\partial \mathbf{k}}(\mathbf{k}) \cdot \delta\mathbf{k} + o\left(\alpha \|\delta\mathbf{k}\|\right) . \tag{17}$$

We set:

$$I_\alpha \;=\; \frac{j(\mathbf{k}+\alpha\,\delta\mathbf{k})-j(\mathbf{k})}{\alpha\;\frac{\partial j}{\partial \mathbf{k}}(\mathbf{k})\cdot\delta\mathbf{k}}\;. \tag{18}$$

According to (17), one must have: $\lim_{\alpha\to 0} I_\alpha = 1$.
The gradient test consists to check this property:

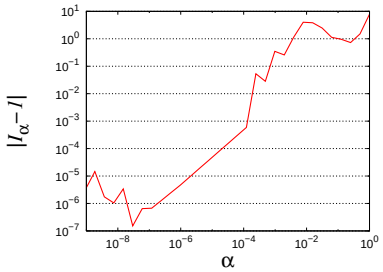
- For an arbitrairy $\mathbf{k}$, we compute $\frac{\partial j}{\partial \mathbf{k}}(\mathbf{k})$ with the adjoint code.
- With the direct code, we compute $j(\mathbf{k})$.
- For $n = 0, \ldots, N$:
  - We compute $\alpha = 2^{-n}$;
  - With the direct code, we compute $j(\mathbf{k}+\alpha\,\delta\mathbf{k})$;
  - We compute I_α ;
- We check if $\lim_{\alpha\to 0} I_\alpha = 1$ or not.

Figure 4.10 (a) shows a result of the gradient test. $|I_\alpha - 1|$ is plotted against α in logarithmic scale. The convergence is good until $\alpha > 10^{-7}$. When α is smaller, the approximation of the partial derivatives is not reliable anymore due to truncation errors, see e.g. [**?**].

5 Benchmarks of the forward code

5.1 Explicit solutions

Academical tests cases with the exact solution known are performed in order to check the finite volume solver (forward code).



(a) Gradient test

```
####################################
##    TEST DU PRODUIT SCALAIRE    ##
####################################

Appel du code linaire tangent...
Appel du code adjoint...
<Xd,Xb> =   -682.277083033688
<Yd,Yb> =   -682.277082428555
relative error :  -8.869324203484993E-010
```

(b) Scalar product test

Figure 4.10: Adjoint code validation

5.1.1 Water at rest

The first test is the water at rest with a variable topography. We check if the finite volume solver is "well balanced" or not, see e.g. [**?**]. That means that it must preserves at least water at rest solutions.

5.1.2 Steady solution and Manning-Strickler formula

This test case aims to check the well known Manning-Strickler formula that links water height, discharge and manning coefficient in the case of a steady and uniform state.

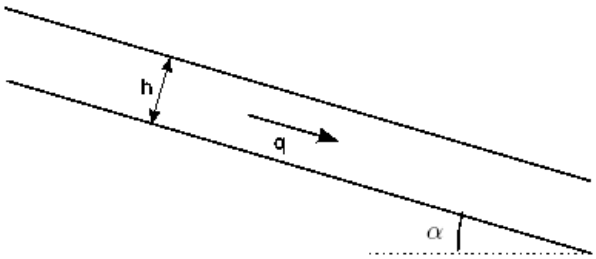


Figure 5.11: Steady state solution for the Manning test case

In the present case (a rectangular and very large channel), the water height h can be computed by:

$$h = q^{3/5}\ n^{3/5}\ L^{-3/5}\ i^{-3/10} \tag{19}$$

where q is the water discharge, n the Manning coefficient, L the canal length and $i = sin(\alpha)$.

This test case consists to compare the steady state solution obtained by DassFlow, to the expression (19).

This test case allows to validate or not the balance between the topography term and the friction term.

5.1.3 Dambreak

This test case simulates a dam-break with a flat bottom and without friction. The domain dimension is $50m \times 10m$, the topography is constant. There is a discontinuity at $x = 25m$. The initial condition is the following :

$$h(x,y) = \begin{cases} 1 & \text{if } x \leq 25 \\ 0.1 & \text{if } x > 25 \end{cases} \text{ see fig. 5.12.}$$

and

$$u(x,y) = \begin{cases} 2.5 & \text{if } x \leq 25 \\ 0 & \text{if } x > 25 \end{cases}$$

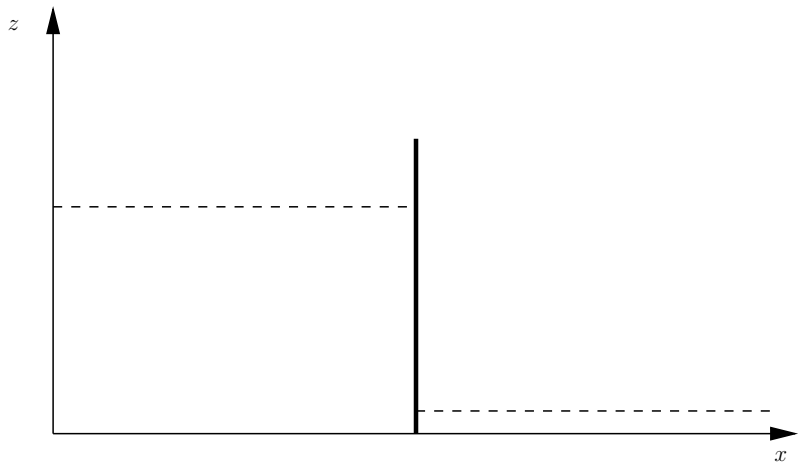


Figure 5.12: Initial condition on h for the dambreak test case

The simulation time is $3s$ and the time step is 0.01. We perform this test case with a regular structured mesh (with $n_x = 200$ and $n_y = 20$) and with an unstructured triangular mesh (based on the same mesh for boundaries). Results are shown in figure 5.13.

5.2 Real data test case: Moselle river

In this section, we present a forward run on real data and compare the results with available measures.
The study concerns the flood event from Feb.25 1997, 12h00 to Mars.2 1997, 12h00 of Moselle river. The mesh has 2340 cells and 2430 nodes, and it is a mix of triangles and quadrangles, see fig. 5.15. It includes the topography defined at the center of cells and was generated using the ArcGIS software. Time step is $dt = 2s$.
This flood event test case is studied in [**?**] and data come from [**?**].

Boundary conditions are imposed using real measures obtained during the flood event. The hydrographs showing the inflow discharge prescribed at inflow boundary and the water elevation prescribed at outflow boundary are shown in figure 5.14.

In order to validate the numerical solution, we compare it to measures available at the EDF gauge station located approximatively in the middle of the main channel, see fig. 5.16. concerning data assimilation experiments conducted on this configuration we refer to [**?**].

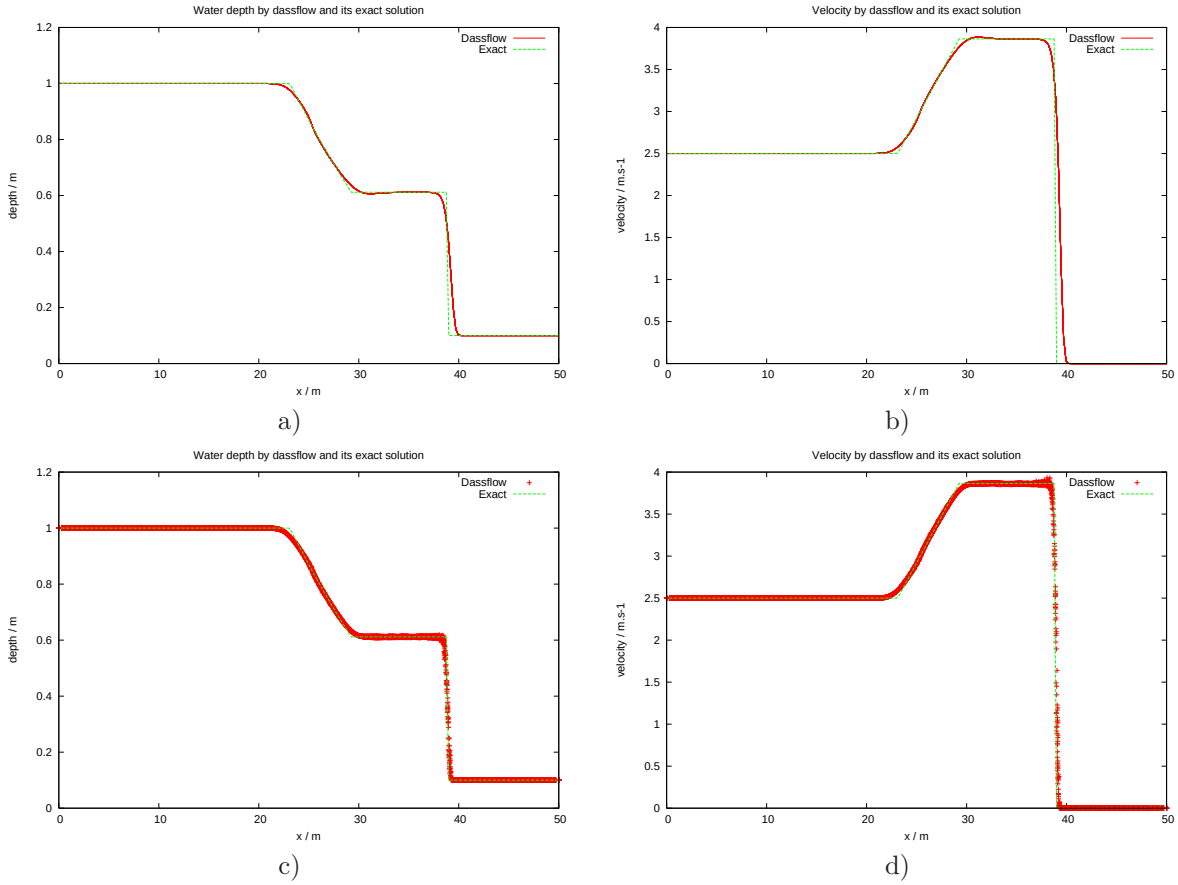
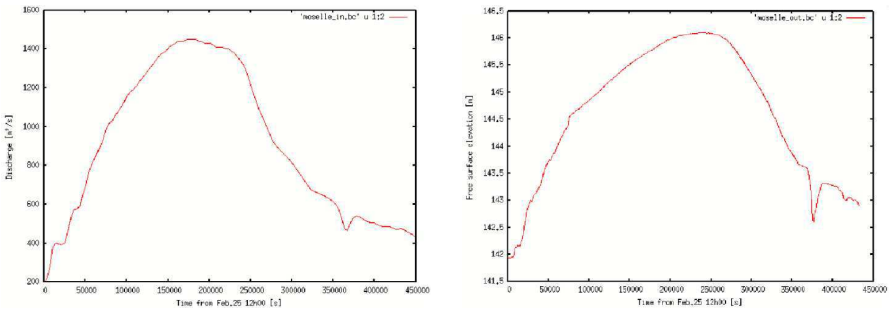


Figure 5.13: Dam break test case (all y values are superposed). a) and b) on a regular structured mesh, c) and d) on an unstructured triangular mesh. a) and c) represent h, b) and d) represent u.

6 Benchmarks for the full code (data assimilation)

In this section, we present some twin experiments results. A "twin experiment" means that a first run of the direct model provides observations. Then control variables are changed (we set them as a "first guess") and we run DassFlow in the data assimilation mode in order to try to retreive the control variable values that generated the observations.



a) b)

Figure 5.14: Moselle river. a) Inflow discharge imposed. b) Elevation prescribed at outflow boundary.

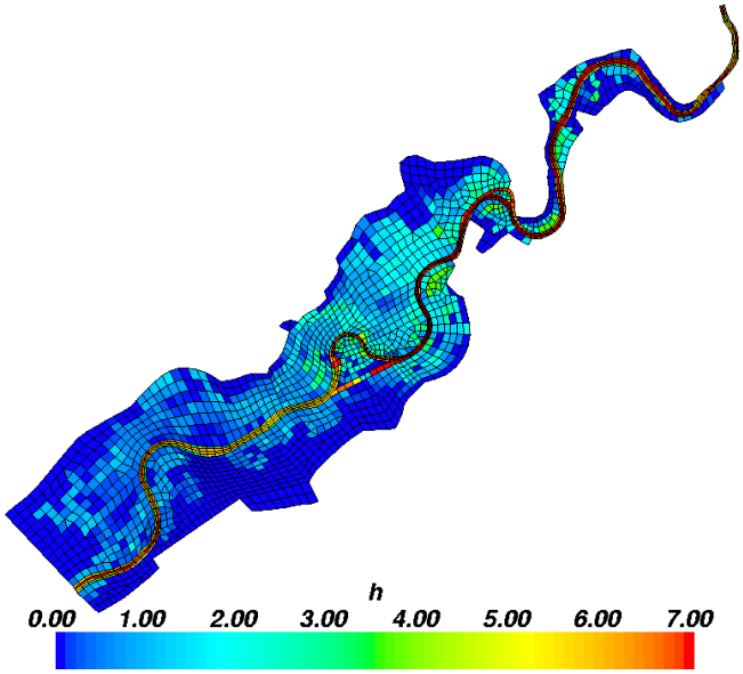


Figure 5.15: Moselle river. Mesh and water depth computed on Feb.26 1997 14h00

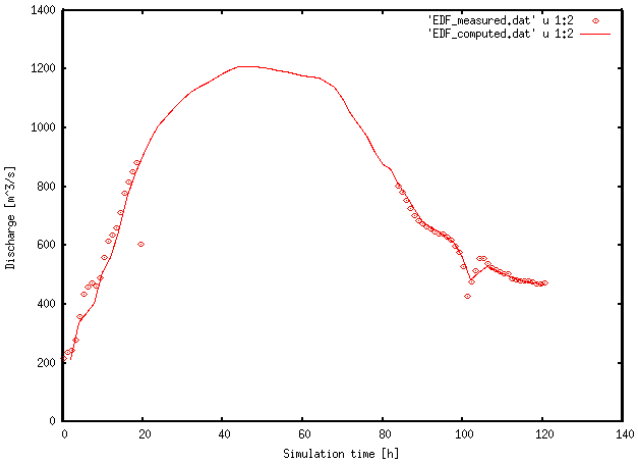


Figure 5.16: Moselle river. Comparison of the measured and computed hydrographs at EDF gauge station

6.1 Identification of a local topography

Our first twin experiment concerns the identification of the topography in a small scale and academic case. The domain is 30 m long and 4 m large, and the topography is defined by :

$$\begin{aligned} z_b(x,y) &= 0.9\exp\left(-\tfrac{1}{4}(x-10)^2\right)\exp\left(-(y-1)^2\right) \\ &+ 0.7\exp\left(-\tfrac{1}{8}(x-20)^2\right)\exp\left(-2(y-3)^2\right) \end{aligned} \tag{20}$$

The inflow boundary is at $x = 0$, the outflow boundary at $x = 30$. Boundaries $y = 0$ and $y = 4$ are walls. We use a rectangular structured mesh of dimension 90×20.
Bed roughness, defined by its Manning coefficient, is uniform ($n = 0.025$). We impose a constant discharge $q^{in} = 8\ m^3/s$ at $x = 0$ and a constant water height $h_{out} = 1.4\ m$ at $x = 30$. We obtain a steady state solution after about 80 s of simulation. Figure 6.17 shows the water height of this steady state solution and the topography.

From this steady state solution, we extract the forthcoming observations: h^{obs} and u^{obs} every $0.02\,s$ during $20\,s$ on each volume. The objective of this test case is to retreive the topography. The first guess used is a flat bottom.
We use the datassim mode, see [**?**], with the following cost function :

$$j_1(z_b) = \frac{1}{2}\int_0^T \left(\left\| h(t) - h^{obs}(t) \right\|_\Omega^2 + \left\| \mathbf{q}(t) - \mathbf{q}^{obs}(t) \right\|_\Omega^2 \right) dt\ , \tag{21}$$

Figure 6.18 shows the cost function and the norm of its gradient normalized by its initial values, vs iterates (a) and the identified topography (b). We can notice that convergence is obtained and the reference topography is well retreived.

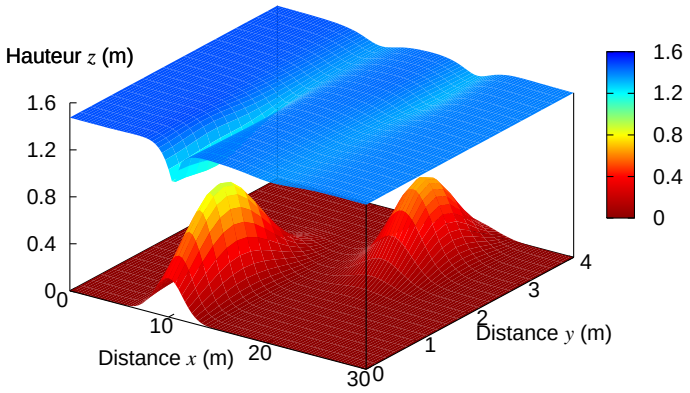
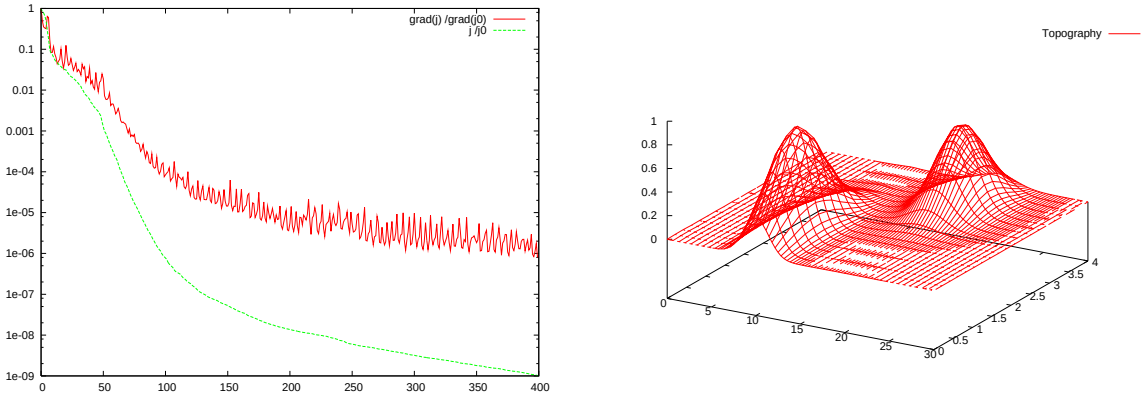


Figure 6.17: Identification of the topography. Topography and steady state elevation.



(a) (b)

Figure 6.18: Value of the cost function and of the norm of its gradient, normalized by their initial values (a) and the identified topography (b)

Further similar assimilation experiments but using lagrangian floaters are conducted in [**?**].

6.2 Identification of inflow discharge

We consider a toy test case which includes many features of real cases (eg. Moselle river). The computational domain contains a main channel (river) and floodplains, see figures 6.19 and 6.20).

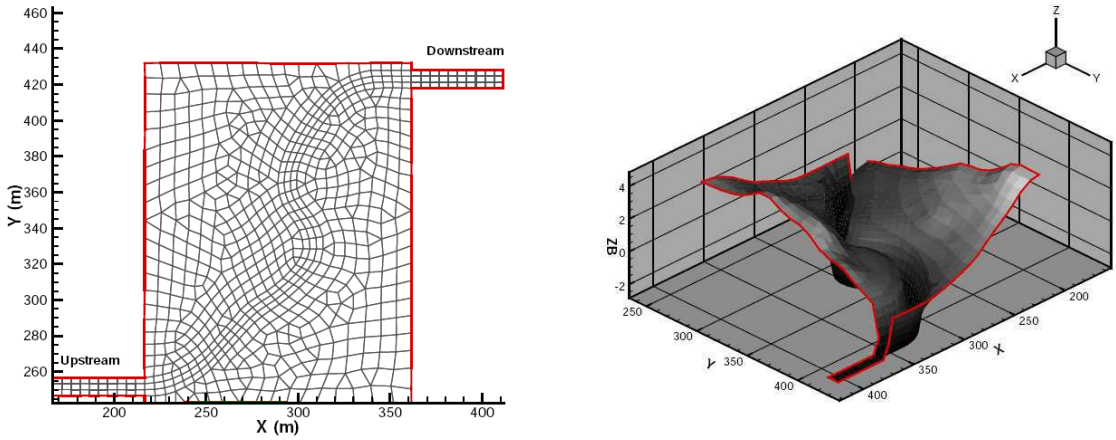
Again, the present test case is a twin experiment. At the inflow boundary, we set the inflow discharge shown if figure 6.21 (a) simulating a flood event.

Then we perform a forward run to generate observations at points 1 and 2 shown with black stars in figure 6.20(a).

Then, we suppose that the inflow discharge is constant ($4.95\, m^3 s^{-1}$), and we try to retreive its real value by assimilating observations.

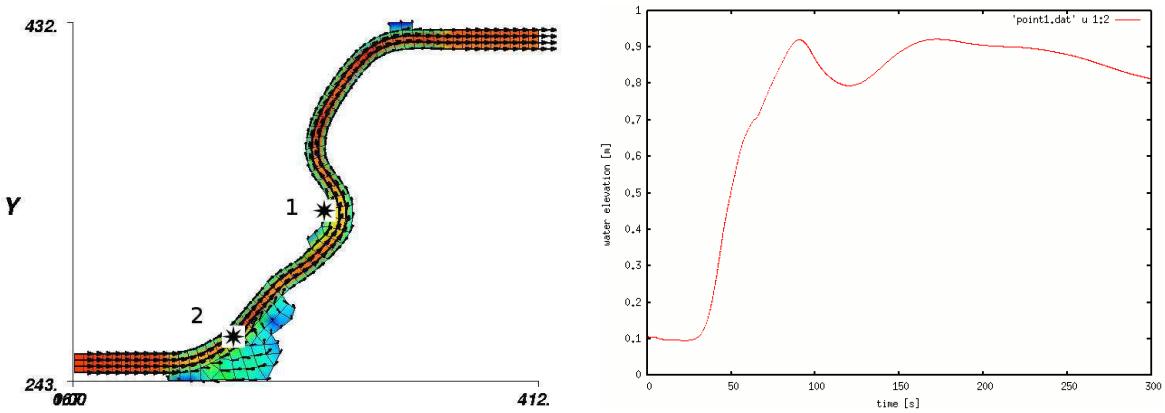
We present in Figure 6.21 the identified inflow discharge for different experiments. In Fig. 6.21(a), observations are h and $\mathbf{q}$ at each cell and each time step. In Fig. 6.21(b), observations are h at point 1 and (h, $\mathbf{q}$) at point 2, both at each time step. In Fig. 6.21(c), observations are h at point 1 only, but at each time step.

We can notice that the identified inflow discharge is good even with the observation of h at point 1 only. Also, we can notice that the end of the flood event is not well identified. This is the "blind period" phenomena: for example in case (c), the inflow discharge after 270 s can not be identified because the information from the inflow boundary did not reach yet the gauge station.



(a) (b)

Figure 6.19: Toy test case mesh (a) and bathymetry (b)



(a) (b)

Figure 6.20: Toy test case domain with mesurment points (a) and observation data available at point 1 (b)

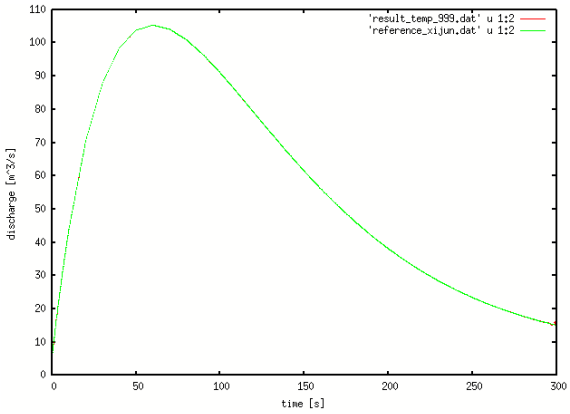
6.3 Identification of Manning coefficients

We consider the same toy test case presented above but we seek to identify manning coefficients in given land-use. Manning coefficients are defined using five different land uses as follows, see figure 6.22:

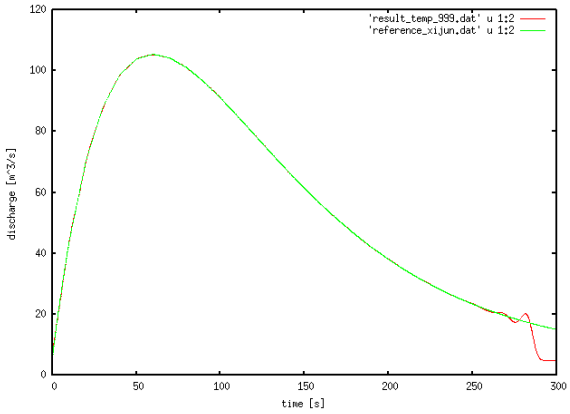
- **In the main channel :** $n = 0.025$ (gravelly main channel)
- **Right to the main channel :** $n = 0.066$ (flood plain with bushes)
- **Left to the main channel :** $n = 0.04$ (flood plain with little vegetation)
- **Near gauging station number 1 :** $n = 0.03$ (pasture, farmland)
- **Near outflow boundary condition :** $n = 0.10$ (urban)

Again, this test case is a twin experiment: a forward run generates the observations. For the present test case, the observations are water height at any time at gauge station 1, see figure 6.20(a).
As first guess we consider a constant Manning ($n = 0.010$). The data assimilation process allows to recover perfectly the values used to generate the observations:

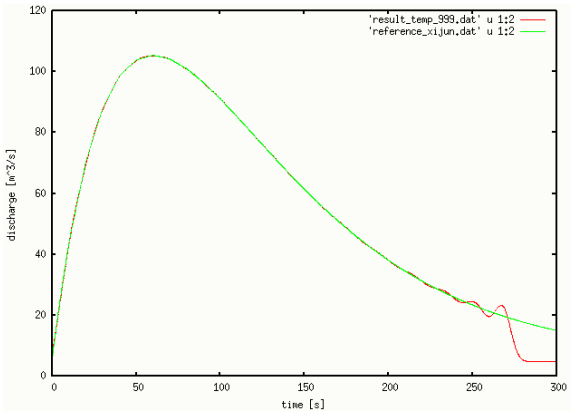
Area	Manning value
Main channel	0.025000487
Flood plain (right)	0.065991635
Flood plain (left)	0.039996868
Near gauging station 1	0.029999658
Near outflow	0.10001384



(a) Observations everywhere



(b) 2 observation points



(c) 1 observation point

Figure 6.21: In green : reference inflow discharge. In red : identified inflow discharge. (a): Observation of $(h, \mathbf{q})$ everywhere; (b): Observation of h at point 1 and $(h, \mathbf{q})$ at point 2. (c): Observation of h at point 1 only.

Extra numerical experiments with real data (Moselle river, see [**?**]), show the efficiency of the present approach.

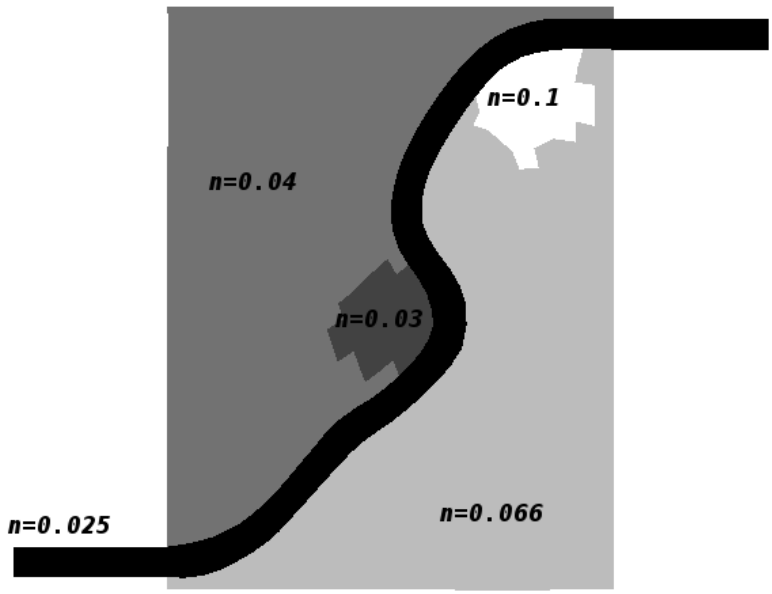


Figure 6.22: Manning coefficients

A Characteristics boundary conditions

In this section, we explain how we define the open boundary conditions based on the theory of characteristics. We refer to [**?**] and [**?**].
Let us consider (1) near the boundary, in the non-conservative form and linearized around a mean value (h_0, u_0, v_0), with a flat bottom and without friction :

$$\begin{cases} \frac{\partial u}{\partial t} + u_0 \frac{\partial u}{\partial x} + v_0 \frac{\partial u}{\partial y} g \frac{\partial h}{\partial x} = 0 \\ \frac{\partial v}{\partial t} + u_0 \frac{\partial v}{\partial x} + v_0 \frac{\partial v}{\partial y} + g \frac{\partial h}{\partial y} = 0 \\ \frac{\partial h}{\partial t} + u_0 \frac{\partial h}{\partial x} + v_0 \frac{\partial h}{\partial y} + h_0 \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) = 0 \end{cases} \tag{22}$$

In a matrix form, this gives:

$$U_t \, + \, A_1 \, U_x \, + \, A_2 \, U_y = \, 0 \tag{23}$$

with $U = [h, u, v]^T$, $A_1 = \begin{pmatrix} u_0 & 0 & g \\ 0 & u_0 & 0 \\ h_0 & 0 & u_0 \end{pmatrix}$, and $A_2 = \begin{pmatrix} v_0 & 0 & 0 \\ 0 & v_0 & g \\ 0 & h_0 & v_0 \end{pmatrix}$

Let $\mathbf{n} = [n_1, n_2]^T$ and τ be respectively the normal and the tangent vector to the boundary (see Fig A.23). The matrix $A = n_1 \, A_1 \, + \, n_2 \, A_2$ has 3 eigenvectors: $w_1 = \mathbf{u} \cdot \mathbf{n} + \sqrt{\frac{g}{h_0}} h$, $w_2 = \mathbf{u} \cdot \tau$ and $w_3 = \mathbf{u} \cdot \mathbf{n} - \sqrt{\frac{g}{h_0}} h$. They are associated to the eigenvalues $\lambda_1 = \mathbf{u}_0 \cdot \mathbf{n} + c$, $\lambda_2 = \mathbf{u}_0 \cdot \tau$ and $\lambda_3 = \mathbf{u}_0 \cdot \mathbf{n} - c$ $(c = \sqrt{g h_0})$ respectively. These eigenvectors w_1, w_2 and w_3 are the so-called **characteristic variables**.

We rewrite (23), using w_1, w_2 and w_3:

$$\begin{cases} \frac{\partial w_3}{\partial t} + \lambda_3 \frac{\partial w_3}{\partial x_{\mathbf{n}}} + \mathbf{u}_0 \cdot \tau \frac{\partial w_3}{\partial x_\tau} - c \frac{\partial v}{\partial x_\tau} = 0 \\ \frac{\partial w_2}{\partial t} + \lambda_2 \frac{\partial w_2}{\partial x_{\mathbf{n}}} + \mathbf{u}_0 \cdot \tau \frac{\partial w_2}{\partial x_\tau} + \frac{c}{2} \frac{\partial (w_1 - w_3)}{\partial x_\tau} = 0 \\ \frac{\partial w_1}{\partial t} + \lambda_1 \frac{\partial w_1}{\partial x_{\mathbf{n}}} + \mathbf{u}_0 \cdot \tau \frac{\partial w_1}{\partial x_\tau} - c \frac{\partial v}{\partial x_\tau} = 0 \end{cases} \tag{24}$$

If we neglect the variations along x_τ, (24) becomes a system of transport equations of w_k at speed λ_k in the normal direction $\mathbf{n}$. Given an open boundary, w_k is incoming if $\lambda_k < 0$ and outgoing otherwise.

Then, the characteristics boundary conditions are implemented as follows:

- For each point of the boundary, we compute the eigenvalues λ_1, λ_2 and λ_3.
- For each λ_k,
  - if $\lambda_k < 0$, then w_k is incoming and must be specified from external data.
  - if $\lambda_k \geq 0$, then w_k is outgoing and must be computed from internal data.

- Then the state variables can be recovered from characteristics values by:
  - $\mathbf{u} \cdot \mathbf{n}^{bnd} = \frac{w_1+w_3}{2}$
  - $\mathbf{u} \cdot \tau^{bnd} = w_2$
  - $h^{bnd} = \sqrt{\frac{h_0}{g}} \frac{w_1-w_3}{2}$

The remaining question is to compute the outgoing characteristics using internal data. Since the finite volume solver is explicit, at current time step n, we need values of the state variable in ghosts cells at time step $n-1$, see section 4.1.4. This is done using a simple linear extrapolation in the normal direction of the normal, see figure A.23.

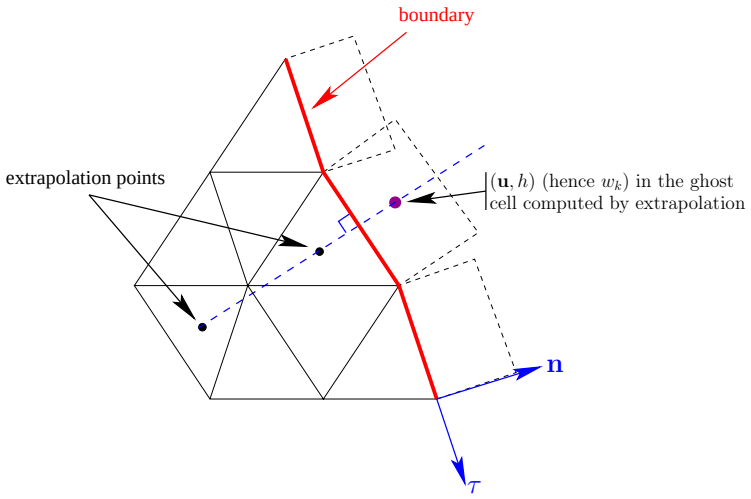


Figure A.23: Computation of characteristic variables in a ghost cell using a linear extrapolation of the values in cells (black points).

B Riemann invariants and boundary conditions

This section details why the relation $\dfrac{\partial}{\partial \mathbf{n}}(\mathbf{u} \cdot \mathbf{n} + 2c) = 0$ is imposed if either water elevation or a rating curve is prescribed. We refer to ([**?**], section 1.3).

Let us consider the 1D shallow water equations with flat topography and without source term:

$$U_t + B\,U_x = 0 \tag{25}$$

with $U = [h, U]^T$, $B = \begin{pmatrix} U & A/b \\ g & U \end{pmatrix}$, and $b(x)$ is the canal width.
The matrix B has 2 eigenvalues $\lambda_1 = U + \sqrt{\frac{g\,A}{b}}$ and $\lambda_2 = U - \sqrt{\frac{g\,A}{b}}$. The corresponding left eigenvectors are $\Lambda_1 = \left[g,\, \sqrt{\frac{g\,A}{b}}\right]^T$ and $\Lambda_2 = \left[g,\, -\sqrt{\frac{g\,A}{b}}\right]^T$. Then (25) gives by linear combination:

$$g\left(\frac{\partial h}{\partial t} + \lambda_1 \frac{\partial h}{\partial x}\right) + \sqrt{\frac{g\,A}{b}}\left(\frac{\partial U}{\partial t} + \lambda_1 \frac{\partial U}{\partial x}\right)$$

or

$$\frac{dU}{dt} + \frac{g}{\sqrt{\frac{gA}{b}}}\frac{dh}{dt} = 0$$

where $\frac{d}{dt} = \frac{\partial}{\partial t} + \lambda_1 \frac{\partial}{\partial x}$ represents the Lagrangian derivative in the direction of the curve of slope λ_1.

The idea is to transform this equation in the form :

$$\frac{d}{dt}(U + \omega) = 0 \tag{26}$$

where ω is a quantity to be determined. Since $\sqrt{\frac{gA}{b}}$ is not constant, we have to solve :

$$d\omega = \sqrt{\frac{gb}{A}}dh \tag{27}$$

In case of a rectangular channel, we obtain: $w = 2\sqrt{gh}$.
This means that $(U + 2\sqrt{gh})$ and $(U - 2\sqrt{gh})$ are conserved along characteristic curves of direction λ_1 and λ_2 respectively.

For an outflow boundary ($\mathbf{u} \cdot \mathbf{n} > 0$), $\lambda_1 > 0$ and $(U + 2\sqrt{gh})$ must be computed from the interior of the domain. Since the present finite volume solver is explicit in time, we need values of the state variable on the boundary at time $n-1$ when computing those of time n. Then, we use a simple extrapolation in the direction of the normal to compute the quantity $(U + 2\sqrt{gh})$ on the boundary.
Using a 0-order extrapolation, we find the well-known condition :

$$\frac{\partial(U + 2\sqrt{gh})}{\partial n} = 0$$

Link with the characteristics boundary conditions

One can notice that the transported quantities obtained are different from those of appendix A. This can be explained by the use of linearized equations in appendix A.

Let us linearize (26) around (h_0, U_0) : $\begin{cases} h(x,t) = h_0 + h'(x,t) \\ U(x,t) = U_0 + U'(x,t) \end{cases}$

We get: $\omega = 2\sqrt{gh_0} + \sqrt{\frac{g}{h_0}}h' + o(h')$. By setting in (26), we get:

$$\frac{d}{dt}(u_0 + 2\sqrt{gh_0}) + \frac{d}{dt}\left(u' + \sqrt{\frac{g}{h_0}}h'\right) + \frac{d}{dt}(o(h'))$$

Assuming h' and U' are small, we find the expression of section A :

$$\frac{d}{dt}\left(u' + \sqrt{\frac{g}{h_0}}h'\right) = 0$$

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399